AFRL-RI-RS-TR-2012-054

FORMAL MODELS OF COMPOSABLE SECURITY ARCHITECTURES

FEBRUARY 2012

FINAL TECHNICAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2012-054 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/

STEVEN T. JOHNS, Chief
Trusted Systems Branch

/s/

PAUL ANTONIK, Technical Advisor
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
FEB 2012	Final Technical Report	OCT 2008 – SEP 2011

4. TITLE AND SUBTITLE

FORMAL MODELS OF COMPOSABLE SECURITY ARCHITECTURES

5a. CONTRACT NUMBER
In House

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)

Dilia E. Rodriguez

5d. PROJECT NUMBER
23T4

5e. TASK NUMBER
PR

5f. WORK UNIT NUMBER
OJ

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITA
525 Brooks Road
Rome, NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site
26 Electronic Parkway
Rome NY 13441

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2012-054

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2012-0310
Date Cleared: 19 JAN 2012

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Much of the research and practice in security is concerned with particular enforcement mechanisms, and implementation or code-level vulnerabilities. This research takes an information-flow approach, which is implementation-independent, and applies it to the specification and analysis of security properties of component-based architectures. The goal was to develop rigorous but lightweight formal support for the development of secure systems. The developed formal models and inference systems are rigorous because their underlying foundation is Mantel's compositional framework for information-flow security, and they are specified in Maude, which is based on rewriting logic, a general yet simple logic of concurrent change. They are lightweight because they are object-based, and automatically generate proofs induced by pattern-based queries. They can be used to explore the design space of systems prior to implementation.

15. SUBJECT TERMS

formal methods, security architectures, information-flow security

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			DILIA E. RODRIGUEZ
U	U	U	UU	20	19b. TELEPHONE NUMBER (*Include area code*) N/A

Contents

1 SUMMARY

Security requirements are varied and complex, and critical to many systems. Much of the research and practice of security is concerned with particular enforcement mechanisms, and implementation or code-level vulnerabilities. This research complements those approaches by taking an information-flow approach, which is implementation-independent, and applying it to the specification and analysis of information-flow security properties of component-based architectures. The goal was to develop rigorous, yet lightweight, formal methods to support the development of secure systems.

The developed formal models and inference systems are rigorous because they have as the underlying foundation on security Mantel's modular compositional framework for information-flow security, and they are specified in the Maude language, which is based on rewriting logic, a general yet simple logic of concurrent change. They are lightweight because they are object-based, and automatically generate proofs induced by pattern-based queries. With them it is possible to explore the design space of applications or systems prior to implementation.

Two classes of formal models for security architectures were developed. With both, the deduction of the global security properties of the architecture proceeds by the transformation of the given model for a component-based architecture into that of a single composite component, according to Mantel's compositional results on composition of information-flow properties, which would have the global security properties the architecture satisfies.

The first is object-based, and models a component as a configuration of objects that represent the satisfaction of elementary information-flow security properties called Basic Security Predicates (BSPs). There is one class for each kind of BSP. With this kind of model it is easy to explore the security properties of a design. Various optimizations to reduce the state-space explosion problem were investigated, including parameterized specifications tailored to an application.

The second class of models is also object-based, and addresses the state-space explosion problem in a more fundamental way. A model is layered: one layer for the architecture of the underlying component-based architecture, and one layer for the architecture or structure of the corresponding security properties. It is a graph with inter- and intra-layer edges. It is also hierarchical: objects that model components of each layer have as attributes architectures that implement them. This serves two purposes. First, such

a component documents global properties of the nested architecture. Second, it offers a natural and significant way to reduce the search space of the deduction process by effectively removing the nested architecture from the distributed state.

This research is a contribution towards formal methods to support the development of secure systems. Some of the techniques developed in this work could be applied to develop modular and efficient formal models in other kinds of applications.

2 INTRODUCTION

Security requirements are critical for many systems, and they should be considered as early as possible in their design. However, presently much research and practice in security is concerned with particular enforcement mechanisms, and implementation or code-level vulnerabilities. These approaches necessarily have to be introduced in late stages of the development of a system. By then many security flaws are difficult to detect and fix. The research in this effort complements those approaches, and is based on the information-flow approach to security. By imposing constraints on the flow of information it is possible to express confidentiality and integrity requirements. While the information-flow approach has been used in language-based security techniques, since it provides implementation-independent guarantees it can also provide a basis for the analysis of security aspects of the architecture of a system.

In [1, 2] Goguen and Meseguer laid the formal foundation for this approach. They introduced the notion of *noninterference*. Intuitively, A is noninterfering with B if the actions of A have no effect on the observations of B. This provides confidentiality for A, and integrity for B. Over the years variants of noninterference have been introduced for various technical reasons. They all have the same basic intuition, and collectively are known as information-flow properties. Whenever they hold, many subtle attacks that exploit programs using legitimate access to data simply are ruled out.

Information-flow properties do not fall within the domain of safety and liveness properties. These are properties of single traces, while information-flow properties are closure properties of sets of traces. Therefore, theories of composition of safety and liveness properties do not apply to information-flow properties. In general, these are not preserved under composition. However, disparate treatments have shown that some information-flow properties, or restrictions of them, do compose [3, 4, 5, 6, 7]. Mantel has proposed a unifying treatment of information-flow properties [8, 9], in which these previous results can be rederived, and which also makes possible the definition of new information-flow properties, and the derivation of their compositional properties.

With Mantel's security framework as the theoretical foundation on security, we have developed models and techniques for the specification and analysis of information-flow properties of component-based architectures. The goal was to develop lightweight, rigorous formal methods that could be used

to explore the design space of secure systems.

Our formalization is based on the Maude system [10, 11], which is a formal, executable language and set of tools based on rewriting logic [12, 13]. This is a general, but simple, logic of concurrent change that has been shown to be a good logical and semantic framework [14]. Among its appealing features in formalizing logics and systems is that it does not prescribe a particular syntax. So one can define the most simple and natural syntax for the particular logic or system of interest.

A logic or concurrent system is specified by a *rewrite theory* $\mathcal{R} = (\Sigma, E, R)$, in which (Σ, E) is an equational theory that specifies the states of the system, and R is a set of *rewrite rules* of the form $t \rightarrow t'$ that specify basic concurrent transitions that transform fragments of a state.

In this work the objective was to investigate and develop suitable inference systems for the analysis of information-flow properties of component-based architectures. An inference system is formalized as a rewrite theory in which the equational theory (Σ, E) specifies a component-based architecture, with the security properties each component satisfies, and the rewrite rules rules R capture results from the security framework proposed by Mantel. In designing a secure system the question is whether secure components yield a secure system. For the inference systems we have developed, executing the specification of a component-based architecture deduces properties of composites of the components. A secure system would be represented by a single composite component that would include all the components of the architecture, along with the information-flow properties it satisfies.

A designer of a component-based architecture might want to check whether it satisfies one or more information-flow properties. In the formal systems we have developed, pattern-based queries can be used to explore the properties of an architecture. Given the specification of an architecture and a query, the system would automatically generate the proofs needed to answer the query. Should an architecture not satisfy desired properties, a designer could make modifications, and again query the system. Thus, the models and techniques we have developed support the exploration of the design space of security architectures. Furthermore, though the security underpinnings and the formalizations are rigorous, the specifications are object-based and easy to understand, and the proofs are generated automatically from queries. Our models and techniques thus provide lightweight, rigorous, formal support for the design of secure systems.

3 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Information-Flow Security

A component or system is formalized in Mantel's framework as an event system $ES = (E, I, O, Tr)$, where E is a set of events, $I, O \subseteq E$ are the sets of input and output events, respectively, and $Tr \subseteq E^*$ is the set of traces or finite sequences over E. A security policy is concerned with a set of security domains, for example, for the intruder, and for one or more different kinds of legitimate user of the system. For each domain, the set of events E is partitioned into three sets according to how information flows to that domain: the set V of events that are directly *visible* from that domain, the set C of events that are *confidential* or kept secret from that domain, and the set N of remaining events, which are *neither*, but which may be deducible. Such a triple of sets $\nu = \langle V, N, C \rangle$ is called a *view*.

Basic Security Predicates (BSPs) [8, 9] are elementary information-flow properties. They are closure properties parametric on views. Given a trace of the system, they require that other traces be among the behaviors of the system, Tr. Each BSP requires enough possible traces to prevent an adversary from deducing information of a particular kind. It is defined in terms of the sets of the view, and some have other subsets of E as additional parameters. For some, the required traces are constructed from given ones by inserting events in some prescribed way: $BSI, BSIA^{\rho}, FCI^{\nabla,\Delta,\Upsilon}$. For others, it is by deleting events from the given trace that the required one is obtained: $R, BSD, FCD^{\nabla,\Delta,\Upsilon}$.

Information-flow properties can be defined in terms of views and a security predicate, where a security predicate defines a closure property that guarantees in some technical sense the information flow specified by a view. A security predicate can be defined as the conjunction of one or more BSPs. Security properties known from the literature have been defined by such conjuctions. For example, separability is defined as $BSD_{\nu_L^{LH}}(Tr) \wedge BSIA^{\rho_C}_{\nu_L^{LH}}(Tr)$. That is, separability requires that the set of traces Tr be closed under two elementary closure properties $(BSD, BSIA^{\rho_C})$ defined in terms of the view ν_L^{LH}, which is the view for the domain of low-level events, L, in a flow policy that has another domain, H, of high-level events.

3.2 An Object-Based Model of Security Architectures

We developed an inference system that given a specification of a component-based architecture, along with the security properties each component satisfies, deduces the global security properties the architecture satisfies, if any. Each component is represented by one object for each BSP it satisfies.

For example, suppose there is a component $X = (E, I, O, Tr)$ that satisfies the separability property defined above, that is, it satisfies $BSD_{\nu_L^{LH}}(Tr)$ and $BSIA^{\rho_C}_{\nu_L^{LH}}(Tr)$. This component is specified by the following objects of classes `BSD` and `BSIA`.

```
< X : BSD |                          < X : BSIA |
    interface :  [e E, i I, o O],        interface :  [e E, i I, o O],
    view :  [v V, n N, c C] >            view :  [v V, n N, c C],
                                         rho-view :  R >
```

These objects represent assumptions that component `X` satisfies *BSD* and $BSIA^{\rho_C}$ in view `[v V, n N, c C]` $= \nu_L^{LH}$, where the value of function ρ_C applied to the view is the set of events `R`.

Theorems in [9] define an ordering on BSPs by implication, under what conditions BSPs satisfied by components are preserved under composition, and under what conditions BSPs are trivially satisfied. These theorems are formalized in Maude as rewrite rules of the form:

$$l : t \rightarrow t' \text{ if } cond$$

where t and t' are configurations of objects denoting security properties of components. If the condition *cond* is true the security property represented by t implies the security property represented by t'. Proofs are constructed by the application of these rules.

There is a class for each BSP, which are subclasses of class `BasicSecProp`, which is a subclass of `SecurityProp`. New subclasses of `SecurityProp` can be introduced for security properties defined by conjunction of BSPs.

Given the specification of a component-based security architecture, it is possible to deduce further properties that the components satisfy, and properties that various composites of the components satisfy. Using Maude's `search` command to query the system about security properties of interest, the system would automatically generate proofs by the application of the rewrite rules that formalize the theorems in [9], and it would return the one or more solutions requested, or report that no solution to the query was found.

For example, suppose we have two components, with identifiers `X` and `Y`, respectively. The specification of the component-based architecture would include one object for each BSP that each of these components satisfy. In general, information-flow properties are not preserved by composition. In our model the satisfaction of security properties by components is represented by objects in the configuration that describes the architecture. A composite of components `X` and `Y` has identifier `X . Y`. We can use this identifier to investigate which properties, if any, this composite satisfies.

If it satisfied some BSP the configuration would include some object of a class corresponding to a BSP that would have the identifier `X . Y`. Then to learn whether it satisfies any BSP, we would submit the following query:

```
search architecture =>*
    [ < X . Y : Prop:BasicSecProp | Atts:AttributeSet >
      C:Configuration ] .
```

The term `architecture` defines the component-based architecture, a configuration that includes `X` and `Y`, and equations that identify output events of one connected to input events of the other. The pattern in the query has an object with id `X . Y` of some subclass of `BasicSecProp`, represented here by the variable `Prop`, which takes as value the name of any subclass of `BasicSecProp`. Failure to find a solution indicates that the composition of `X` and `Y` satisfies no BSP, and thus no security property.

Should this query not fail, we could investigate further the security aspects of the composite. Information-flow properties are satisfied in particular views, which define how information flows to some domain, for example the intruder, or some kind of legitimate user. The views of the components induce a set of possible views for systems composed from them. A query could specify a particular view, or a pattern for it. As in the following example:

```
search architecture =>*
    [ < X . Y : BSD |
        view :  [v V:EventSet, n N:EventSet , c c12],
        Atts:AttributeSet >
      C:Configuration ] .
```

which seeks to determine whether the composite satisfies the basic security predicate *BSD* in some view that keeps the events in the set `c12` secret.

An information-flow property can be defined in terms of BSPs, and a new subclass of `SecurityProp` can be introduced for it. Then queries could check for the satisfaction of this new property. With these kinds of queries it is possible to explore the design space for some application or system.

3.3 A Modular and Reflective Model of Security Architectures

Though the model described above supports the exploration of the design space, it suffers from the state-space explosion problem. Various optimizations ameliorated the problem, but the search for more fundamental and effective solutions led to the model described in this section.

Separation of concerns suggests that the specification of the architecture of a system be layered: one layer for the underlying components and how they are connected, and another for the security properties of the components and how they are composed. For each a formal object-based specification is natural. The relative notion of component suggests further elaboration of the model. A component may be very simple and atomic, or may be composed of other simpler components. Thus, it is natural to introduce hierarchical models.

The purpose of the architectural models we developed was to investigate techniques to derive their global properties from the properties of their components. This is achieved by the transformation of the architecture into a single composite component whose properties would be global properties of the given architecture. In a model in which the architecture has a base layer and a security layer, the transformation of the base layer is accomplished through operations derived from the composition of event systems, while the transformation of the security layer is accomplished through transitions derived from Mantel's compositional results. The base layer is concerned with the sets of events of its components. The security layer is concerned with the security properties of the base components. The connection between the two layers comes through views: a view partitions the set of events of a component, and a security property holds for particular views.

Base Layer. The base layer represents a component-based architecture as a configuration of objects that is of sort (or type) `ArchConf`. Each component is represented by an object of class `Component`, and connections between a pair components are represented explicitly by a single object of class `Connection`.

```
sorts    ComponentSet ConnectionSet ArchConf .
subsorts ComponentSet ConnectionSet < ArchConf < Configuration .
```

Components may be very simple and atomic, or may be implemented by some architecture. Thus, a `Component` object has two attributes: `interface`

and `implementation`, which is a term of sort `ArchConf`, but "frozen", not subject to transitions.

The configuration that models an architecture represents a graph in which the edges are defined by the identifiers of the objects. For example, given components with ids `X` and `Y` there may be at most one connection between them, whose id would be `< X, Y >`. This defines edges between connection `< X, Y >` and component `X`, and between connection `< X, Y >` and component `Y`. This connection specifies how output events of component `X` are connected to input events of `Y`, if any, and how output events of component `Y` are connected to input events of `X`, if any. These three objects specify a composition that leads to the creation of an object of class `Component` with id `< Y, X >`. As part of the transformation induced by this composition these objects migrate from the main configuration to be nested as the value of the `implementation` attribute of the new `Component`, where they are no longer subject to rewrites or transformations. Furthermore, the architectural invariant is reestablished: every connection must connect exactly two components in that configuration, and there may be at most one connection between any two components.

That the composite component nests the architectural configuration that induced its composition serves two purposes. First, it documents that the nested configuration has the interface `[e E, i I, o O]` given in that object. As we consider in the security layer the security properties of this nested component-based architecture, the relevant views will be ones that partition this set `E` of events. Second, since the nested configuration is no longer subject to rewrites, the reduction of the search space for the application of rewrite rules results in more efficient transformation techniques.

The ids of objects in the architectural configuration are structured, and can become deeply nested as sequences of compositions are performed. A lexicographic order on ids is defined, and for any connection `< X, Y >`, id `X` precedes id `Y` in that ordering. Ids not only serve to define the edges of the graph, but the id of an object encodes the sequence of transformations that led to its creation.

Security Layer. The base layer consists of components and connections. The security layer has corresponding elements: some that represent the security properties a component satisfies, and others that impose constraints on the composition of security properties of connected components. The

security layer is connected with the base layer through views.

Given the security properties of the base components, and constraints on the connections between them, the goal is to deduce global security properties of the security architecture. These follow from the compositionality results for BSPs. These are parametric on views, and there are some constraints on views of components whose security properties would be composable. First, whenever an output of one component is connected to an input of another, the two events are identified. This means that views for these components must agree on these events. Second, given views $\nu_1 = \langle V_1, N_1, C_1 \rangle$ and $\nu_2 = \langle V_2, N_2, C_2 \rangle$, any composition of BSPs that hold in these views requires that N_1 and N_2 be disjoint.

Then every connection in the base layer imposes constraints on the views of the connected components. Therefore corresponding to `Connection` objects in the base layer there are objects of class `ViewConnection` in the security layer. They have the following form:

```
< [ < X, Y >, N ] : ViewConnection |
    v :  < e1-x, e1-y > .  .  .  < en-x, en-y >,
    c :  < e1-x', ed1-y' > .  .  .  < em-x', em-y' >,
    Atts >
```

A `ViewConnection` defines subviews for matching events of connected base components. Matching events in the `v` attribute are assigned to be events in the visible component of a view, while those in the `c` attribute are assigned to be confidential.

A `Component` object has a single interface, a triple `[e E, i I, o O]`, comprising the set of all events, and the sets of input and output events of the component. There may be, however, more than one view to partition `E`. For each view ν one or more BSPs may hold; for example, BSI_ν and R_ν. Statements about what security properties hold in a view are represented as objects of the class `ViewSecProp`. Consider the following example:

```
< [1 : 2] :  ViewSecProp |
    c-view :  CV,
    u-view :  UV,
    properties :  BSI | R,
    Atts >
```

This is a statement about a view ν_2 for the base component `1`. There are other views for this component, at least ν_0 and ν_1. Each view is partitioned into a view for its connected events, `c-view`, and a view for the unconnected

events, `u-view`. This object states that BSI and R hold in ν_2, that is, BSI_{ν_2} and R_{ν_2}.

In the base layer a composition is specified by a connection `< X, Y >` and the two components it connects, `X` and `Y`. In the security layer a corresponding composition is much more complex, but it involves a `ViewConnection` with id `[< X, Y >, Nl]` and `ViewSecProp` objects with ids `[X : Nl1]` and `[Y : Nl2]`. While composition in the base layer creates a `Component` object with id `< Y, X >`, one of the corresponding compositions in the security layer results in a `ViewSecProp` object with id `[< Y, X > : Nl']`, which gives security properties that the composite component `< Y, X >` satisfies.

As the model described in the previous section, this model can support the exploration of the design space of applications and systems. In addition, since it is layered, graph-based, and hierarchical it better captures the structure of the architecture, and relations between various views and security properties. Furthermore, it makes possible some optimization techniques in a simple and natural way.

4 RESULTS AND DISCUSSION

We developed models and inference systems for the analysis of information-flow security properties of component-based architectures.

A first class of models is object-based, and represents components of the architecture as configurations of objects of classes that represent BSPs. This formalization supports the exploration of the design space of applications or systems by pattern-based queries. Several models were developed to ameliorate the state-space explosion problem. They included parameterized specifications that could be tailored for an application.

A second class of models, also object-based, is layered, graph-based, and hierarchical. A base layer captures the structure of the underlying component-based architecture. A security layer captures the structure of the corresponding information-flow security properties. This formalization also supports the exploration of a design space. In addition, the hierarchical nature of both layers offered a natural and significant way to reduce the search space of the graph-transformation techniques.

5 CONCLUSIONS

This research is a contribution towards formal methods to support the development of secure systems. It complements language-based methods and implementation-dependent techniques that can be applied only late in the development process. The models and inference systems we developed offer formal support for the exploration of the design space of component-based architectures. The specification and analysis of architectures is possible because this work is based on the information-flow approach to security, which is implementation-independent. The formal methods developed are rigorous, yet lightweight. They are rigorous due to their underlying foundation on security: Mantel's modular framework for the composition of information-flow properties, and to their formalization in the Maude, a language and system based on rewriting logic, which is a general, yet simple, logic of concurrent change. They are lightweight due to their object-based formalization, and automated generation of proofs induced by pattern-based queries. Some of the techniques developed in this work could be of benefit in other kinds of applications.

6 REFERENCES

References

[1] Joseph A. Goguen and José Meseguer. Security Policies and Security Models. In *IEEE Symposium on Security and Privacy*, pp 11–20, 1982.

[2] Joseph A. Goguen and José Meseguer. Inference Control and Unwinding. In *IEEE Symposium on Security and Privacy*, pp 75–86, 1984.

[3] D. McCullough. Specifications for Multi-Level Security and a Hook-Up Property. In *IEEE Symposium on Security and Privacy*, pp 161–166, 1987.

[4] D. McCullough. A Hookup Theorem for Multilevel Security. In *IEEE Transactions on Software Engineering*,**16(60**, 1990.

[5] J. McLean. A General Theory of Composition for Trace Sets Closed under Selective Interleaving Functions. In *IEEE Symposium on Research in Security and Privacy*, pages 79–93, 1994

[6] J. McLean. A General Theory of Composition for a Class of "Possibilistic" Security Properties. *IEEE Transactions on Software Engineering*, 22(1):53–67, 1996.

[7] A. Zakinthinos and E. S. Lee. A General Theory of Security Properties. In *IEEE Symposium on Security and Privacy*, pages 94–102, 1997.

[8] Heiko Mantel. Possibilistic Definitions of Security — An Assembly Kit. In *IEEE Computer Security Foundations Workshop*, pages 185–199, May 2000.

[9] Heiko Mantel. On the composition of secure system. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 88–101, May 2002.

[10] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude — A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*. Springer, 2007.

[11] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. Specification and Programming in Rewriting Logic. Theoretical Computer Science, **285**, no. 2, Elsevier, 2002.

[12] José Meseguer. Conditional rewriting logic as a unified model of concurrency. Theoretical Computer Science 96(1) (1992) pp. 73–155.

[13] Roberto Bruni and José Meseguer. Generalized rewrite theories.1 In *Proceedings of ICALP'03*, in Lecture Notes in Computer Science, vol. 2719, 2003, pp. 252–266.

[14] Narciso Martí-Oliet and José Meseguer. Rewriting logic as a logical and semantic framework. In: Gabbay, D.M., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. 9, 2nd. edn., pp. 1–87. Kluwer Academic Publishers, Dordrecht (2002)

[15] Narciso Martí-Oliet and José Meseguer. General logics and logical frameworks. In: Gabbay, D. M. (ed.) What is a Logical System? Studies in Logic and Computation, vol. 4, pp. 355–392. Oxford University Press, Oxford (1994)

LIST OF ACRONYMS

- BSP — Basic Security Predicate